Membuat Form Data

Pada bab ini kita akan membuat form untuk pemasukan data CD. Susunan pembahasan di sini diatur agar mudah diikuti dan dicari kembali. Pada kenyataannya, kita melakukan pembuatan program berdasarkan komponen apa yang sudah dibutuhkan.

6.1 Membuat frm_Data

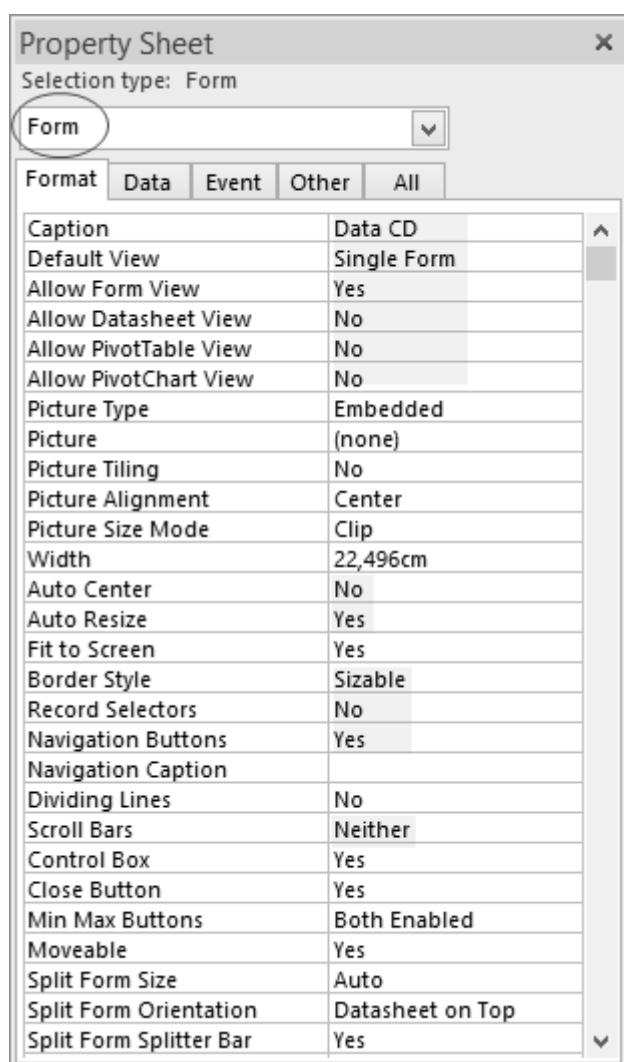
Form **frm_Data** adalah form untuk memasukkan data CD, dan sekaligus untuk menampilkan data. Form ini akan kita buat **pop-up**, artinya akan menjadi jendela (*window*) di atas jendela Access.

Untuk membuat form, klik ribbon tab **Create**, klik "**Form Design**".

Simpan form dengan nama **frm_Data**.

TIP: Sebaiknya langsung menyimpan form walaupun belum diisi apa-apa. Dalam pembuatan form (dan juga objek lainnya) kita akan sering melakukan penyimpanan, dan biasanya cukup dengan klik ikon disket pada **Quick Access Toolbar** atau menekan **Ctrl+S**.

6.1.1 Mengatur Properti Form

Sekarang, tekan **F4** (atau klik ribbon tab **Design**, klik "**Property Sheet**") untuk menampilkan **Property Sheet**, lalu aturlah properti form sebagai berikut.

Gambar 6-1. Property Sheet

Properti-properti yang perlu dipastikan/diatur nilainya adalah sebagai berikut (properti yang lain biarkan dalam keadaan aslinya).

Caption	Data CD
Default View	Single Form
Allow Form View	Yes
Allow Datasheet View	No
Allow PivotTable View	No
Allow PivotChart View	No
Auto Center	No
Auto Resize	Yes
Border Style	Sizable
Record Selectors	No
Navigation Buttons	Yes
Scroll Bars	Neither
Record Source	Tbl_CD
Pop Up	Yes
Modal	Yes
Cycle	Current Record

Berikut ini beberapa properti yang perlu dijelaskan:

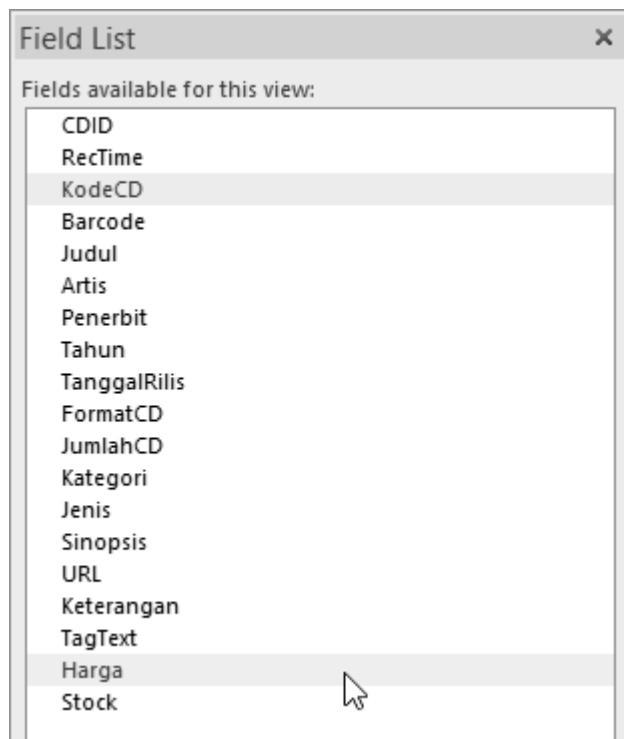
- Jika Anda mengatur properti **Caption** sebelum menyimpan form, maka properti tersebut akan diubah dengan nama form ketika Anda menyimpannya! Ini mungkin '*bug*' pada software Access, maka saya menganjurkan untuk menyimpan form terlebih dahulu ☺.
- **View** yang dibolehkan di sini hanya "Form View", karena kalau tidak dibatasi maka user bisa mengubah form ke view yang lain sehingga akan tampak berantakan.

- **Border Style** kita gunakan "Sizable" agar (jika diinginkan) pemakai masih bisa membesarkan form. Form mungkin perlu dilebarkan oleh pemakai agar *datasheet* yang memuat isi CD bisa dilebarkan (nanti kita akan menggunakan trik pada event **Resize**).
- **Record Selectors** tidak perlu ditampilkan pada form seperti ini. Selain hanya mengambil ruang, juga kurang enak dilihat (subjektif ya?).
- **Navigation Buttons** diaktifkan agar pemakai (yaitu data-admin) bisa mem-*browse* record satu demi satu. Pada pemakaian form untuk menampilkan data (yaitu modul untuk pengunjung), Navigation Buttons ini akan disembunyikan.
- **Scroll Bars** tidak diaktifkan karena pada form ini semua *control* akan terlihat (kita memang membuat form dalam bentuk *window*).
- Saat ini **Record Source** kita gunakan langsung dari table "tbl_CD" agar kita bisa mengambil field-field di dalamnya.
- **Pop Up** digunakan untuk membuat form berbentuk jendela (*window*).
- **Modal** digunakan agar pemakai tidak bisa mengklik di luar form (sebelum form ditutup). Properti ini tidak terlalu penting juga karena kalau pun pemakai mengklik di luar form, tidak akan 'membahayakan' form atau data.
- **Cycle** diisi dengan "Current Record" agar pemakai tidak pindah ke record yang lain ketika menekan Tab pada field terakhir dalam urutan Tab Order. Properti ini pun tidak terlalu penting untuk diatur karena –nanti– kita yang akan 'mengatur' apakah pemakai bisa pindah ke record lain atau tidak.

Catatan: kedua properti yang terakhir di atas (**Modal** dan **Cycle**), saya cantumkan karena umumnya termasuk properti yang diatur pada pembuatan form "pop up".

Selain pengaturan properti, pada form terdapat juga **event**. Kita akan memasukkan *event procedure* ini nanti setelah dibutuhkan.

6.1.2 Menambahkan Control

Sekarang kita akan menambahkan field-field dan control yang diperlukan ke atas form. Klik ribbon tab **Design**, lalu klik "**Add Existing Fields**" untuk menampilkan daftar field (yang berasal dari table yang dijadikan Record Source).

Gambar 6-2. Field List

Klik field yang ingin ditambahkan lalu seret (*drag*) ke atas form. Anda bisa menggunakan **Ctrl+Click** atau **Shift+Click** untuk menyeleksi banyak field sekaligus, untuk ditambahkan. Atur tata letak field sehingga tampak sebagai berikut.

Gambar 6-3. Mengatur letak field

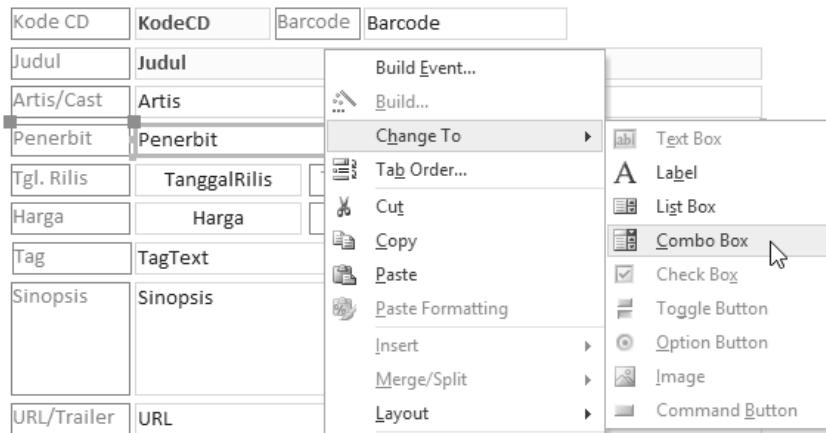
Seperti terlihat pada gambar di atas, pada bagian kiri-atas form, saya menambahkan sebuah control **Image**. Control ini untuk menampilkan gambar sampul (*cover*) CD. Ketika menambahkan control **Image**, pilihlah sebuah gambar (apa saja) yang kecil dulu agar control ini tercipta. Setelah itu Anda bisa menghapus properti **Picture**-nya sehingga menjadi 'kosong' kembali. Properti **Border Style**-nya diatur menjadi "Solid", lalu beri nama (properti **Name**): **Image1**.

Setelah itu tambahkan sebuah control **Label** dengan **Caption** = "*double-click untuk memasang gambar*". Label ini hanya untuk memberi tahu petugas entri data bagaimana memasang gambar sampul CD. Label diletakkan di tengah-tengah control **Image**, lalu "Send To Back" (klik ribbon tab **Arrange**, klik "**Send To Back**") agar Label berada di belakang control **Image**.

Selanjutnya Anda bisa memberi warna-warna untuk menarik perhatian pemakai pada field yang penting, misalnya warna merah pada field **KodeCD** dan **Judul**.

Untuk field **Penerbit**, ubah control-nya menjadi **Combo Box**. Di sini kita bisa membantu petugas entri data dengan menyediakan pilihan penerbit (*publisher*) CD, sehingga petugas tidak perlu mengetik data yang sama berulang-ulang.

Demikian juga untuk field **FormatCD**, **Kategori**, dan **Jenis**, ubahlah menjadi Combo Box. Field-field ini akan menampilkan pilihan dari data master.

Untuk mengubah control menjadi Combo Box, klik-kanan pada control tersebut lalu pilih "**Change To > Combo Box**" (ingat kembali pelajaran ini di Microsoft Access – Basic ya ☺).

Gambar 6-4. Mengubah field menjadi Combo Box

Catatan: saya tidak memberikan penjelasan lebih rinci bagaimana mengatur tata letak control, karena hal tersebut merupakan keterampilan dari materi sebelumnya.

6.1.3 Mengatur Properti Control

Selanjutnya kita perlu mengatur beberapa properti dari control yang sudah ditambahkan. Harap diperhatikan bahwa **semua control** yang berasal dari field sudah mengandung **nama** (properti **Name**) sesuai nama field.

Untuk beberapa field, yaitu field **TanggalRilis**, **Tahun**, **JumlahCD**, **Harga**, dan **Stock,** saya buat rata tengah (**Text Align = Center**). Demikian juga dengan field **FormatCD**, **Kategori**, dan **Jenis**, semuanya dibuat rata tengah. Hal ini tentu bukanlah keharusan, namun hanya untuk 'keindahan' (estetika).

Properti control yang perlu Anda atur lagi adalah (properti yang lain biarkan dalam keadaan aslinya):

- Combo Box: **Penerbit**

Row Source	qry_Penerbit
Row Source Type	Table/View/StoredProc
Bound Column	1
Limit To List	No

  Sebenarnya hanya properti **Row Source** saja yang perlu diubah/diisi, karena properti yang lainnya tetap dalam keadaan asli (properti lain dicantumkan hanya untuk memastikan, misalnya jika Anda telanjur mengubahnya).

  Properti **Limit To List = No** digunakan agar pemakai boleh memasukkan data baru (yang tidak tersedia dalam list).

- Combo Box: **FormatCD**

Row Source	tbl_Master_FormatCD

- Combo Box: **Kategori**

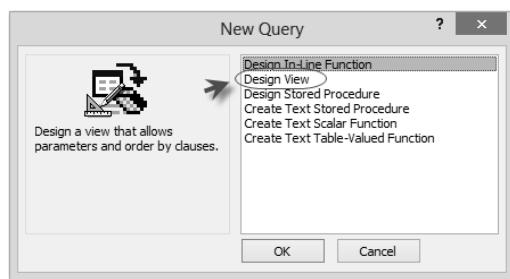
Row Source	qry_Kategori

- Combo Box: **Jenis**

Row Source	qry_Jenis

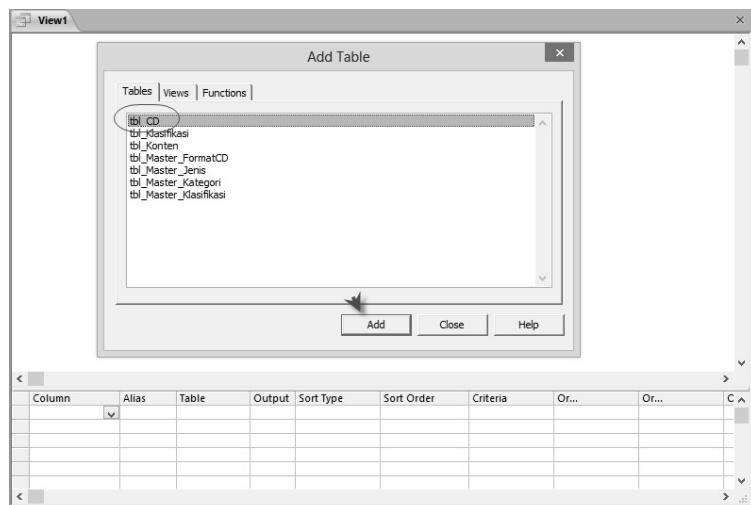
6.1.4 Membuat Query untuk Row Source

Seperti terlihat pada properti control di atas, kita menggunakan beberapa query, yaitu **qry_Penerbit**, **qry_Kategori**, dan **qry_jenis**.

Untuk membuat query, klik ribbon tab **Create**, klik "**Query Wizard**".

Gambar 6-5. Query Wizard

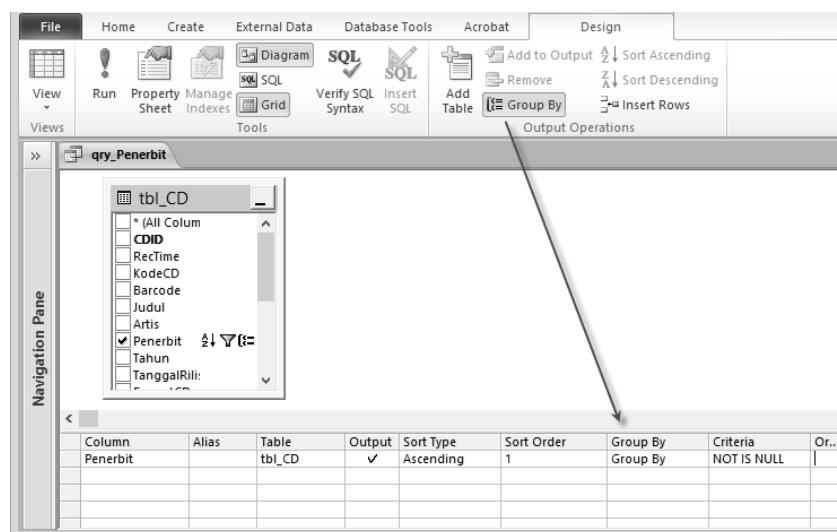
Pada pembuatan query hampir selalu kita gunakan "**Design View**", di mana kita bisa membuat query SELECT seperti yang biasa di Access. Kita menggunakan "Design Stored Procedure" jika ingin membuat query ACTION, yaitu query untuk menambahkan data (INSERT), menghapus data (DELETE), dan mengubah data (UPDATE).

- **qry_Penerbit**

  Sekarang buatlah **qry_Penerbit** dengan **Design View**. Tambahkan **tbl_CD** ke dalam query.

Gambar 6-6. Menambahkan table ke dalam query

Catatan: Jika secara tidak sengaja Anda menutup kotak "Add Table", klik kembali pada ribbon tab **Design**, lalu klik "**Add Table**". Anda bisa juga langsung menyeret (*drag*) table dari **Navigation pane** ke dalam query.

Setelah tbl_CD berada dalam query, tutup kotak "Add Table".

Beri tanda centang pada field **Penerbit** untuk menambahkannya ke dalam *grid* (di bagian bawah). Pada grid, klik kolom **Sort Type** dan pilih "Ascending". Pada kolom **Criteria** masukkan "NOT IS NULL" (tanpa tanda kutip). Terakhir, klik pada ribbon tab **Design**, lalu klik "**Group By**".

Gambar 6-7. Desain query qry_Penerbit

qry_Penerbit akan mengambil data Penerbit yang sudah dimasukkan oleh pemakai, kemudian mengelompokkannya sehingga sebuah penerbit hanya muncul sekali. Dengan cara ini, kita tidak perlu menyediakan table master untuk penerbit, tetapi langsung memanfaatkan data yang dimasukkan oleh pemakai.

- **qry_Kategori**

  qry_Kategori akan mengambil data dari **tbl_Master_Kategori**, ditambah dengan data kategori yang dimasukkan oleh pemakai (kita membolehkan pemakai memasukkan data baru selain memilih data yang sudah kita siapkan). Dengan demikian, query ini merupakan query UNION, yaitu menggabungkan data dari beberapa sumber.

  Untuk membuat **qry_Kategori** kita tidak bisa menggunakan Visual Designer, namun harus mengetik sendiri perintah SQL-nya. Tentu Anda harus mengerti sedikit tentang perintah SQL ini, yang bisa dipelajari misalnya di situs:

  http://www.w3schools.com/sql

  Klik ribbon tab **Create**, klik "**Query Wizard**". Pilih "**Design View**", klik "OK".

  Klik "Close" untuk menutup kotak "Add Table", karena kita akan mengetik sendiri perintah SQL-nya.

  Klik "**SQL**" pada ribbon tab **Design** (untuk menampilkan bidang perintah SQL, di bawah grid).

  Pada bidang SQL tersebut, ketik perintah sebagai berikut:

```
SELECT Kategori
FROM (SELECT Kategori
   FROM tbl_Master_Kategori
   UNION
   SELECT Kategori
   FROM tbl_CD) Src
```

  Klik pada bidang diagram (bagian atas), maka akan muncul "sub-query" bernama **Src** yang merupakan Union Query dari tbl_CD dan tbl_Master_Kategori.

Gambar 6-8. Sub-query yang merupakan Union Query

Sampai di sini, query kita sudah bisa menghasilkan data hasil penggabungan tbl_CD dan tbl_Master_Kategori, namun semua data akan ditampilkan/tidak unik. Agar menjadi unik, klik "**Group By**" pada ribbon tab **Design**.

Selanjutnya, pada kolom **Criteria** masukkan "NOT IS NULL" (tanpa tanda kutip). Kriteria "NOT IS NULL" ini digunakan agar data yang ditampilkan pada list tidak termasuk data kosong (*blank*).

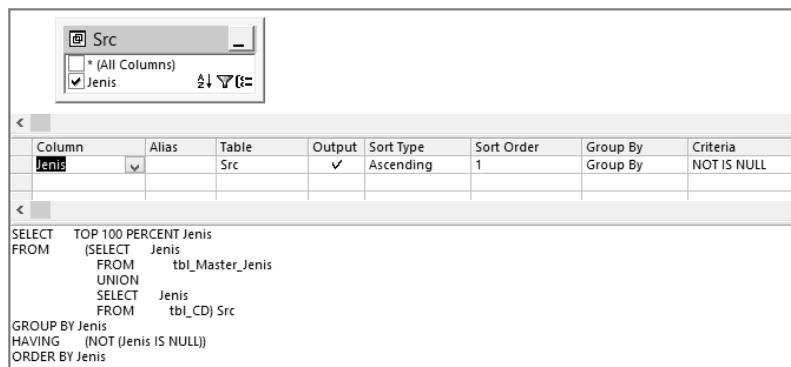
Agar list menjadi urut, pada kolom **Sort Type** pilih "Ascending".

Sekarang query akan tampak sebagai berikut.

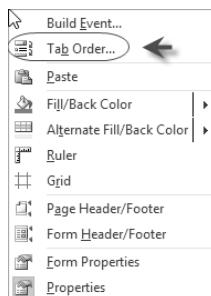
Gambar 6-9. Desain query setelah dimodifikasi

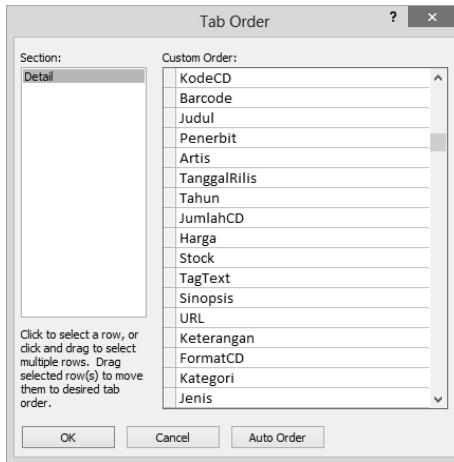
- qry_Jenis

  qry_Jenis hampir sama dengan **qry_Kategori**, yaitu menggabungkan antara data yang kita siapkan pada **tbl_Master_Jenis** dan data yang dimasukkan oleh pemakai pada **tbl_CD**.

  Buatlah **qry_Jenis** dengan cara pembuatan sama dengan **qry_Kategori** di atas, dan hasil akhirnya akan tampak sebagai berikut.

Gambar 6-10. Desain qry_Jenis

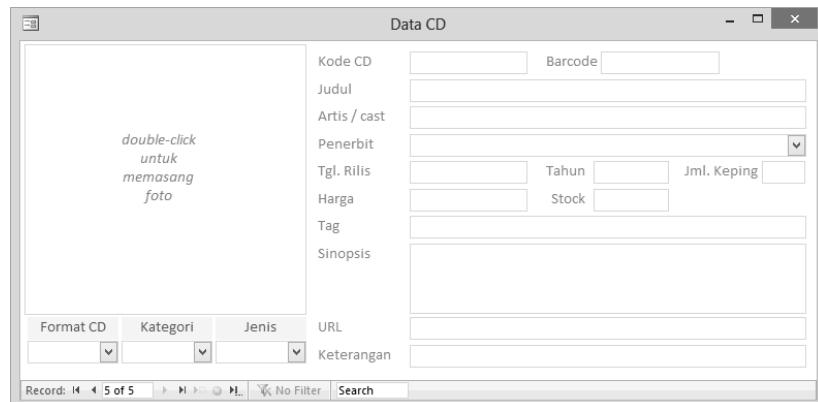
6.1.5 Mengatur Tab Order

Tab Order adalah urut-urutan logis field ketika pemakai menekan tombol **Tab** (di keyboard) untuk pindah field. Untuk menampilkan jendela **Tab Order**, klik-kanan pada *section-bar* (atau bisa juga pada bidang kosong pada form), lalu pilih “Tab Order”.

Gambar 6-11. Menampilkan Tab Order

Pada form yang kita buat, urutan tab-nya sebagai berikut.

Gambar 6-12. Tab Order

Anda bisa memindahkan letak field dengan mengklik field-selector di ujung kiri lalu bawa ke posisi yang diinginkan.

6.1.6 Mencoba Program

Sampai pada tahap ini, form **frm_Data** sudah bisa dicoba untuk dijalankan (**Form View**). Tampilan form akan tampak sebagai berikut.

Gambar 6-13. Tampilan form data

Control combo box **Penerbit**, **FormatCD**, **Kategori**, dan **Jenis** juga seharusnya sudah bisa diklik untuk menampilkan list data (kecuali combo box **Penerbit** karena belum ada data yang dimasukkan). Jika form yang Anda buat belum bisa tampil seperti gambar di atas atau ada *error*, Anda harus menelusuri kembali pembuatan form sesuai petunjuk di atas.

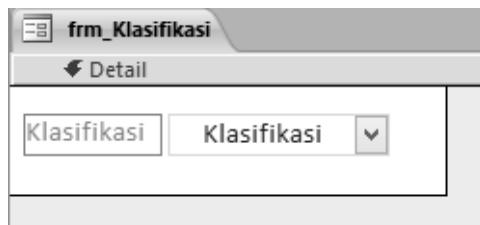
6.2 Membuat frm_Klasifikasi

Jika Anda melihat kembali tampilan form pada Gambar 4-1, di bagian bawah form terdapat dua datasheet. Datasheet yang di sebelah kiri untuk klasifikasi/ genre, dan yang di sebelah kanan untuk isi/konten CD.

Klasifikasi CD dibuat dalam bentuk datasheet karena bisa lebih dari satu. Sebuah CD musik mungkin saja dikelompokkan ke dalam "Dangdut" dan "Campursari" sekaligus karena lagu-lagu di dalamnya ada yang Dangdut dan ada juga yang Campursari (ini hanya misal).

Kita tidak bisa menyediakan satu, dua, atau tiga field pada tbl_CD, misalnya bernama Klasifikasi1, Klasifikasi2, dan Klasifikasi3, karena tidak pasti klasifikasinya lebih dari satu, dan juga jangan dibatasi maksimum tiga. Dalam database, kita mengakomodir keperluan ini dengan menyediakan relasi *one-to-many*.

Sekarang buatlah sebuah form baru dengan **Design View**, lalu simpan dengan nama **frm_Klasifikasi**.

Isi properti Record Source dengan **tbl_Klasifikasi**, lalu tambahkan field **Klasifikasi** ke atas form. Ubah field **Klasifikasi** menjadi combo box.

Gambar 6-14. Desain frm_Klasifikasi

6.2.1 Mengatur Properti Form

Form **frm_Klasifikasi** mempunyai properti sebagai berikut (yang perlu diatur).

Default View	Datasheet
Allow Form View	No
Allow Datasheet View	Yes
Allow PivotTable View	No
Allow PivotChart View	No
Navigation Buttons	No
Record Source	Tbl_Klasifikasi

Untuk Combo box **Klasifikasi**, atur properti-nya sebagai berikut.

Row Source	EXEC sp_Klasifikasi 'Music'

6.2.2 Membuat sp_Klasifikasi

Pada properti di atas kita gunakan **Stored Procedure** bernama **sp_Klasifikasi** dengan parameter 'Music'. Parameter 'Music' ini hanyalah untuk sementara karena nanti parameter ini akan diubah sesuai **Kategori** yang dipilih oleh pemakai.

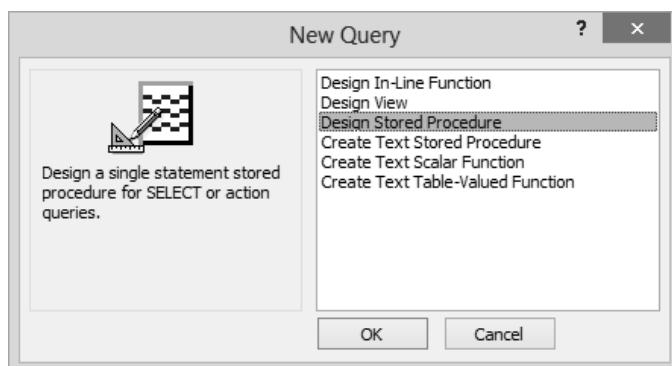
Mengapa menggunakan Stored Procedure? Alasannya di sini karena kita perlu mengirimkan parameter. **View** (query SELECT yang biasa) tidak bisa menerima parameter, sedangkan Stored Procedure bisa menerima parameter.

Stored Procedure **sp_Klasifikasi** sendiri sebenarnya hampir sama dengan **qry_Kategori** atau **qry_Jenis**, yaitu menggabungkan data yang ada di table master dengan data yang dimasukkan oleh pemakai. Dalam hal ini, karena data Klasifikasi disimpan dalam tbl_Klasifikasi, maka **sp_Klasifikasi** menggabungkan data Klasifikasi dari **tbl_Master_Klasifikasi** dan **tbl_Klasifikasi**.

Cara pembuatan **sp_Klasifikasi** pun hampir sama dengan pembuatan qry_Kategori atau qry_Jenis, yaitu dengan menulis terlebih dahulu perintah UNION query untuk dijadikan sumber (sub-query), lalu menambahkan "GROUP BY" agar data yang ditampilkan adalah data yang unik.

Catatan: kalau Anda bertanya lagi mengapa menggunakan **sub-query**, maka alasan saya adalah untuk menghemat jumlah objek (dan panjangnya *step* yang harus dijelaskan, terlepas dari soal optimasi query itu sendiri). Banyak cara yang bisa kita lakukan untuk mencapai sesuatu, dan akhirnya "kebiasaan"lah yang sering memengaruhi cara mana yang digunakan. Kita di sini masih belum membahas cara apa yang paling efisien atau yang paling optimum, karena pada SQL Server sendiri ada lagi yang disebut dengan **Query Execution Plan**, dan lain-lain. Pembahasannya tentu di luar lingkup buku ini.

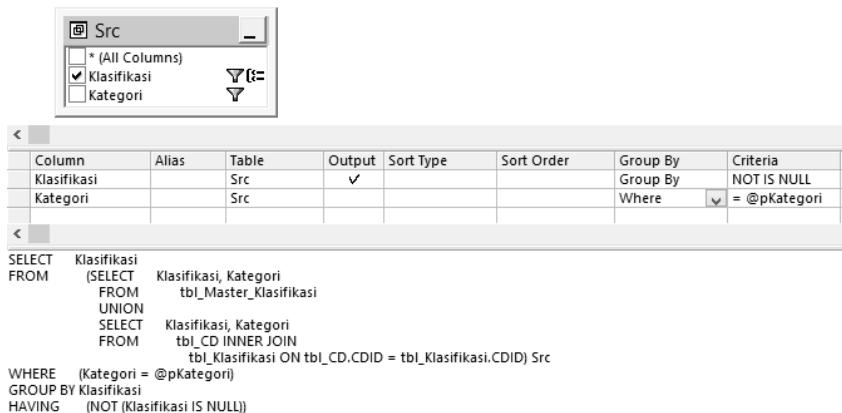
Berikut ini langkah-langkah pembuatan **sp_Klasifikasi**:

1. Pada **Microsoft Access** Project, klik ribbon tab **Create**, klik "**Query Wizards**".

Gambar 6-15. Query Wizards

2. Pilih "**Design Stored Procedure**", klik "**OK**".
3. Klik "**Close**" untuk menutup kotak "Add Table" yang muncul.
4. Klik "**SQL**" pada ribbon tab Design untuk menampilkan bidang SQL (di bawah grid).

5. Ketik perintah berikut ini pada bidang SQL:

```
SELECT Klasifikasi
FROM (SELECT Klasifikasi, Kategori
   FROM tbl_Master_Klasifikasi
   UNION
   SELECT Klasifikasi, Kategori
   FROM tbl_CD INNER JOIN
   tbl_Klasifikasi ON tbl_CD.CDID = tbl_Klasifikasi.CDID) Src
WHERE (Kategori = @pKategori)
GROUP BY Klasifikasi
HAVING (NOT (Klasifikasi IS NULL))
```

Tampak tampilan Stored Procedure sebagai berikut.

Gambar 6-16. Desain sp_Klasifikasi

6. Simpan Stored Procedure dengan nama **sp_Klasifikasi**.

6.2.3 Memberi Hak EXEC sp_Klasifikasi

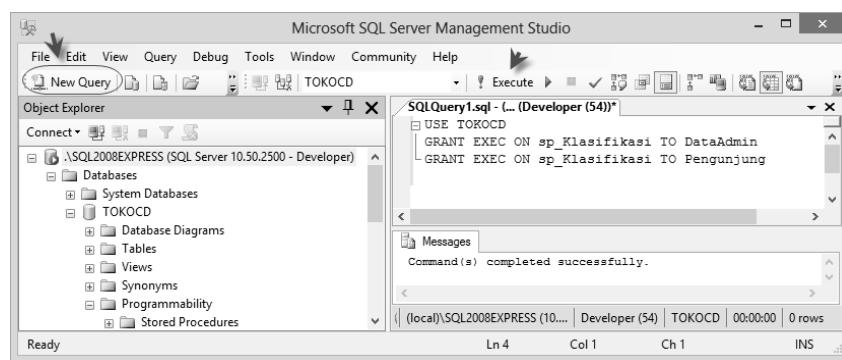
Perlu diketahui bahwa Stored Procedure adalah sebuah objek yang bisa mengandung code/script pemrograman, jadi tidak hanya untuk membaca data. Oleh karena pemakai program (menggunakan login **DataAdmin** dan **Pengunjung**) kita batasi haknya, maka untuk bisa 'menjalankan' Stored Procedure perlu diberikan hak tersendiri.

Cara yang mudah untuk memberi hak tersebut adalah melalui perintah query pada **SQL Server Management Studio**. Langkah-langkahnya sebagai berikut:

1. Jalankan **SQL Server Management Studio** lalu lakukan koneksi ke database **TOKOCD** (gunakan login **Developer** atau **sa**).

2. Klik "**New Query**" pada Toolbar, lalu masukkan perintah berikut:

```
USE TOKOCD
GRANT EXEC ON sp_Klasifikasi TO DataAdmin
GRANT EXEC ON sp_Klasifikasi TO Pengunjung
```

3. Klik "**Execute**" pada Toolbar (atau tekan **F5**) untuk menjalankannya.

Gambar 6-17. Menjalankan query pada SQL Server Management Studio

Catatan: Query yang dibuat di atas bisa diabaikan/tidak usah disimpan karena hanya untuk menjalankan perintah saja.

6.2.4 Membuat Function fRefreshList()

Sekarang tiba saatnya kita sedikit 'bermain' *code*.

Pada pembahasan di atas Anda sudah mencoba memasukkan perintah SQL pada desain View dan Stored Procedure. Kini kita akan menggunakan perintah bahasa VBA (Visual Basic for Application). Tentu Anda tidak asing lagi dengan VBA setelah mempelajari Microsoft Access – Programming.

Function **fRefreshList()** adalah *user-defined function* (UDF). Function ini digunakan untuk me-*refresh* data dalam *list* combo box.

Perlu diketahui bahwa ketika pemakai memasukkan data baru (tidak terdapat dalam list) pada combo box, data baru tersebut tidak langsung tersedia dalam

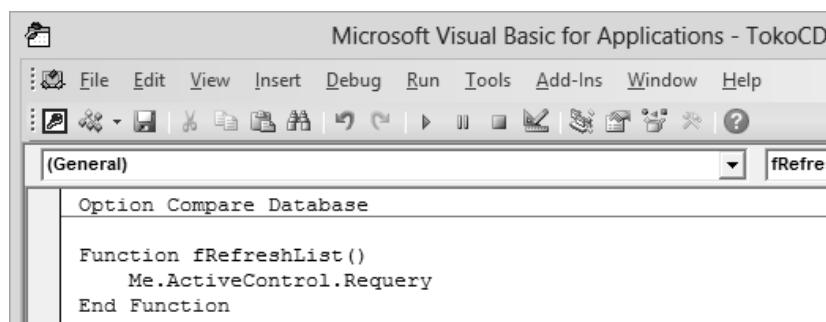
list sehingga tidak bisa dipilih oleh pemakai. Secara Access, pemakai harus menekan **F9** untuk me-refresh list combo box. Tetapi tombol F9 ini tidak populer. Jika tidak menggunakan F9, pemakai harus keluar dulu dari form (menutup form) kemudian menjalankan form kembali, baru data yang dimasukkan tersebut muncul dalam list.

Agar data baru langsung ada dalam list, kita menggunakan sedikit trik, yaitu memanfaatkan event **On Enter**. Event ini terjadi ketika *focus* masuk ke control. Jadi ketika pemakai masuk ke combo box, program otomatis melakukan *requery* untuk memperbarui list.

Mengapa menggunakan *function*, dan tidak langsung melakukan *requery* pada *event procedure* saja? Jawabannya adalah agar kita tidak perlu membuat event procedure untuk setiap combo box. Function bisa langsung dipasang pada properti **On Enter**.

Sekarang mari kita membuat function **fRefreshList()**.

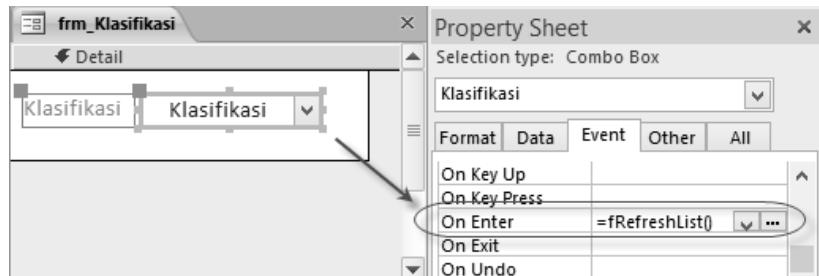
1. Pada **Form Design** (di mana **frm_Klasifikasi** dibuat), klik ribbon tab **Design**, lalu klik "**View Code**". Kita akan meletakkan function **fRefreshList()** pada module form yang bersangkutan.
2. Pada **Visual Basic Editor** yang terbuka, ketik *code* berikut ini:

```
Function fRefreshList()
    Me.ActiveControl.Requery
End Function
```

Gambar 6-18. Membuat function fRefreshList() pada Visual Basic Editor

3. Anda bisa melakukan penyimpanan (**Save**), atau langsung menutup Visual Basic Editor karena code pada module form akan ikut disimpan bersama formnya.

Catatan: nama function ini tentu tidak harus **fRefreshList()**, namun bisa saja seperti **fPerbaruiDaftar()**. Pemakai huruf "**f**" di awal function juga tidak harus, karena huruf tersebut (bagi saya) hanya untuk meng-ingatkan bahwa *procedure* tersebut berupa sebuah *function*, bukan ***sub***. Agar sesuai dengan penyebutan dalam buku ini, sebaiknya Anda memberi nama function sebagai **fRefreshList()**.

Function **fRefreshList()** diletakkan pada *module form* (bukan *module general*) karena mengandung *identifier* "Me" yang merujuk pada objek formnya.

Sekarang pasanglah function **fRefreshList()** pada *event* **On Enter** dari combo box **Klasifikasi**. Gunakan tanda "=" (sama dengan) di awal nama function, sehingga menjadi **=fRefreshList()**.

Gambar 6-19. Memasang function fRefreshList() pada event On Enter

6.3 Membuat frm_Konten

frm_Konten adalah form yang akan kita tampilkan dalam bentuk *datasheet* dan menjadi *sub-form* pada **frm_Data**. Form ini berfungsi untuk menam-pung dan/atau menampilkan data isi CD.

Tata cara pembuatan **frm_Konten** hampir sama dengan pembuatan frm_Klasifikasi. Pertama-tama, simpan form dengan nama **frm_Konten**.

6.3.1 Mengatur Properti Form

Properti form **frm_Konten** adalah sebagai berikut.

Default View	Datasheet
Allow Form View	No
Allow Datasheet View	Yes
Allow PivotTable View	No
Allow PivotChart View	No
Navigation Buttons	No
Record Source	Tbl_Konten

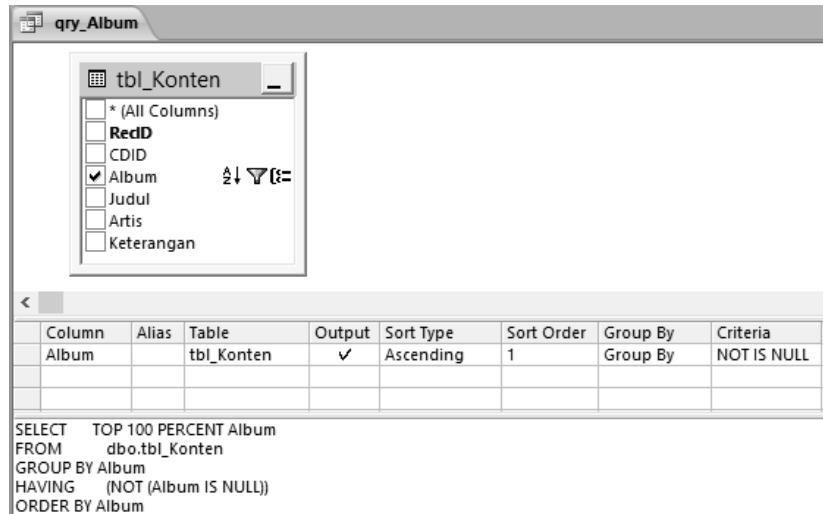
6.3.2 Menambahkan Control

Tambahkan field-field dari **tbl_Konten** yang menjadi **Record Source** form, lalu ubah field **Album** dan **Artis** menjadi Combo Box, sehingga desain **frm_Konten** akan tampak sebagai berikut.

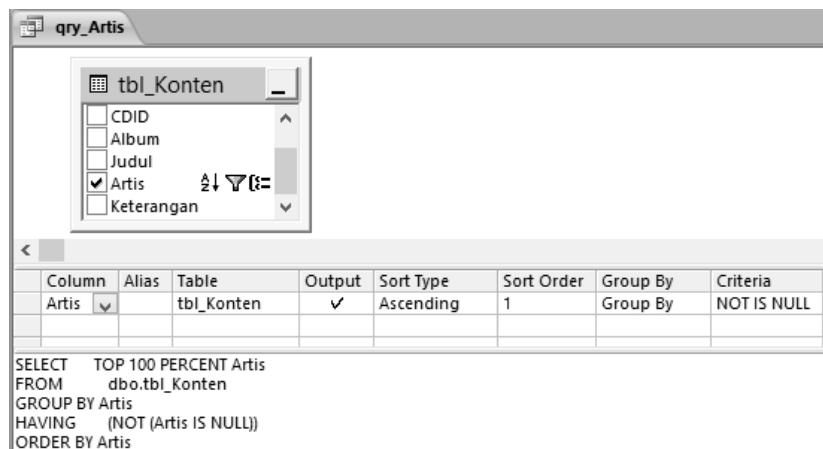
Gambar 6-20. Desain frm_Konten

Untuk combo box **Album** dan **Artis**, data yang ditampilkan dalam list berasal dari data yang pernah dimasukkan pemakai. Combo box ini dibuat untuk membantu petugas entri data agar tidak perlu mengetik ulang nama album atau artis yang sudah pernah dimasukkan.

Properti **Row Source** untuk combo box **Album**: **qry_Album**.

Adapun **qry_Album** bisa dibuat dengan **Query Design View**, sebagai berikut.

Gambar 6-21. Desain qry_Album

Properti **Row Source** untuk combo box **Artis**: **qry_Artis**.

Adapun **qry_Artis** bisa dibuat dengan **Query Design View**, sebagai berikut.

Gambar 6-22. Desain qry_Artis

Untuk combo box **Album** dan **Artis**, jangan lupa menambahkan function **fRefreshList()** pada module form (kopi dari frm_Klasifikasi), lalu pada event **On Enter** masukkan =**fRefreshList()**.

6.3.3 Membuat frm_Konten_1 dan frm_Konten_2

Seperti telah dibahas di bagian depan, isi/konten CD tergantung pada "jenis" CD-nya, apakah merupakan CD "Album" yang berdiri sendiri atau CD "Kompilasi" yang mengandung banyak album.

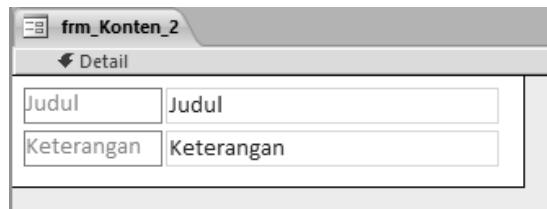
Form **frm_Konten** yang kita buat di atas, cocok untuk merekam data CD yang berupa "Kompilasi", karena di situ terdapat field **Album** dan field **Artis** untuk merekam data album.

Jika **frm_Konten** digunakan untuk merekam data CD berjenis "Album", atau CD "Software", "Games", dan lain-lain, maka field Album dan Artis menjadi tidak bermanfaat. Oleh karena itu, sebaiknya field-field tersebut disembunyikan saja.

Cara yang saya gunakan adalah membuat **frm_Konten_1** untuk merekam data CD "Kompilasi" dan **frm_Konten_2** untuk merekam data CD lainnya.

frm_Konten_1 bisa dibuat langsung dari **frm_Konten**, tinggal mengubah namanya (*rename*) dari **frm_Konten** menjadi **frm_Konten_1**.

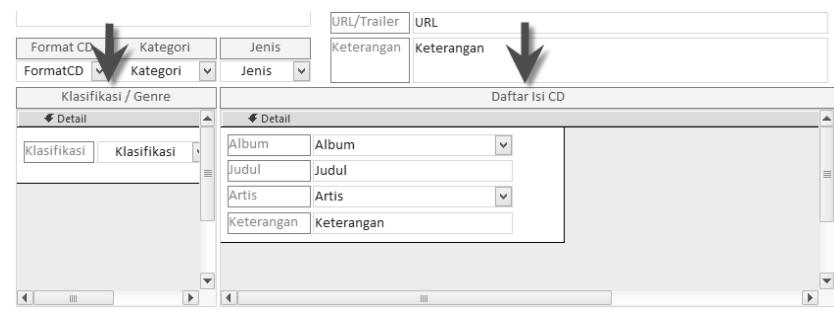
Untuk **frm_Konten_2** dibuat dengan cara "Save As" dari **frm_Konten_1** (buka **frm_Konten_1** lalu klik menu **File > Save Object As**).

Pada **frm_Konten_2**, hapuslah combo box **Album** dan combo box **Artis** sehingga menjadi sebagai berikut.

Gambar 6-23. Desain frm_Konten_2

Oleh karena pada **frm_Konten_2** sudah tidak ada Combo Box yang memerlukan function **fRefreshList()**, maka function tersebut bisa dihapus dari module-nya.

6.4 Menambahkan Subform

Bukalah **frm_Data** ke dalam **Design View** kembali untuk menambahkan datasheet "klasifikasi" dan "konten".

Layout **frm_Data** (bagian bawah) setelah Anda menambahkan dua buah subform akan tampak sebagai berikut.

Gambar 6-24. Subform pada frm_Data

Untuk subform sebelah kiri, atur propertinya sebagai berikut.

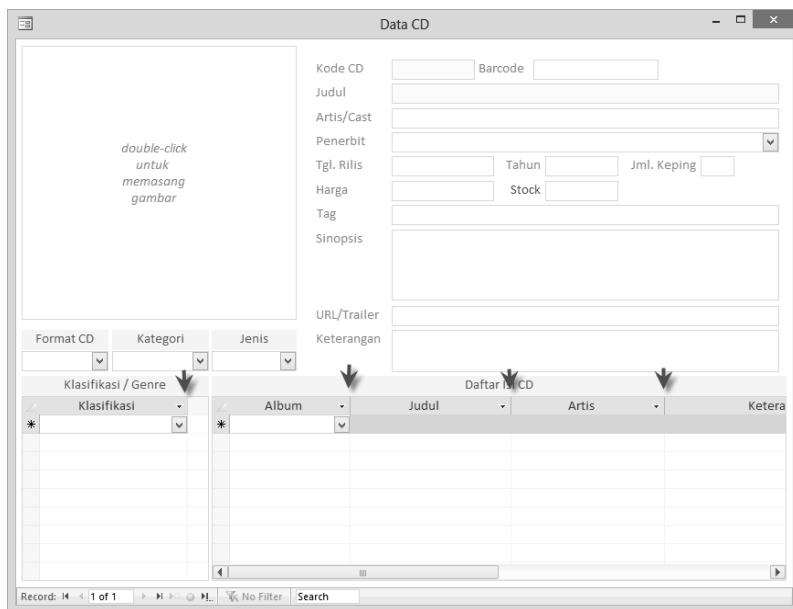
Source Object	frm_Klasifikasi
Link Master Fields	CDID
Link Child Fields	CDID
Name	Sub1

Untuk subform sebelah kanan, atur propertinya sebagai berikut.

Source Object	frm_Konten_1
Link Master Fields	CDID

Link Child Fields	CDID
Name	Sub2

Simpan **frm_Data** lalu tampilkan dalam **Form View**.

Di sini Anda perlu mengatur letak kolom pada datasheet, termasuk lebar kolom sehingga 'enak' dilihat dan memudahkan pemasukan data. Setelah melakukan pengaturan, tekan **Ctrl+S** untuk menyimpan *layout* datasheet (sehingga tampak seperti itu ketika ditampilkan kembali).

Gambar 6-25. Mengatur letak dan ukuran kolom datasheet

6.5 Menambahkan txtDummy

Sampai tahap ini, seperti terlihat pada gambar di atas, **frm_Data** yang Anda buat sudah bisa dijalankan (ditampilkan dalam **Form View**) dan sudah bisa menerima pemasukan data. Semua data yang Anda masukkan akan masuk ke table yang benar.

Perlu diingatkan lagi bahwa form data ini bisa digunakan untuk memasukkan data dan juga menampilkan data yang ada dalam database. Jika Anda perhatikan, setiap kali Anda membuka form dan menampilkan data yang sudah ada, field **KodeCD** akan otomatis terseleksi (*selected*) karena menempati urutan pertama dalam Tab Order. Selain 'mengaburkan' penglihatan terhadap data (data terlihat kurang jelas), data yang terseleksi bisa berisiko terhapus atau diganti secara tidak sengaja. Untuk itu kita perlu mencegah agar *focus* tidak otomatis masuk ke field data. Salah satu cara yang saya gunakan adalah membuat sebuah Text Box *dummy* bernama **txtDummy** dan diletakkan pada urutan pertama dalam Tab Order.

Cara membuat **txtDummy**:

1. Tambahkan sebuah control Text Box ke atas form (bisa di mana saja).
2. Klik Label-nya lalu hapus. Kita tidak memerlukan Labelnya.
3. Klik Text Box tersebut lalu atur propertinya sebagai berikut.

Top	0
Left	0
Width	0
Height	0
Name	txtDummy

   Text box **txtDummy** memang sengaja dibuat tidak kelihatan oleh mata tetapi tetap visible (Visible = Yes) dan bisa menerima tab (Tab Stop = Yes).

4. Atur Tab Order sehingga **txtDummy** berada pada urutan pertama.

Mulai sekarang, setiap kali Anda menampilkan data pada form, field **KodeCD** tidak otomatis mendapat *focus*. Anda harus menekan **Tab** untuk masuk ke field **KodeCD** atau mengklik field tersebut.

6.6 Menambahkan Tombol Simpan dan Undo

Sebenarnya, salah satu 'kelebihan' Microsoft Access dibanding *tool* aplikasi database yang lain adalah kita bisa menggunakan *bound control*, di mana data yang dikandung pada *control* langsung berasal dari database dan setiap update yang dilakukan terhadap data akan otomatis tersimpan pada database. Dengan demikian, kita tidak memerlukan mekanisme penyimpanan data sendiri. Begitu pindah record (termasuk menutup form) maka data otomatis akan tersimpan.

Namun demikian, pemakai aplikasi database tidak mengetahui hal ini. Walaupun kita memberitahunya, kebiasaan orang dalam memasukkan atau meng-update data adalah menyimpan data tersebut (melakukan penyimpanan). Untuk mengakomodasi kebiasaan ini, kita bisa menyediakan tombol "Simpan". Walaupun tombol ini lebih ke sifat artifisial, namun bisa juga kita berikan fungsi yang sebenarnya, yaitu untuk menyimpan data.

Di samping kebutuhan untuk menyimpan data, muncul pula kebutuhan untuk membatalkan (perubahan) data, atau yang disebut "Undo". Jadi, selain menyediakan tombol "Simpan", kita perlu juga menyediakan tombol "Undo".

Berikut ini cara membuat kedua tombol tersebut:

1. Pastikan Anda tidak mengaktifkan "Use Control Wizards", atau jika nanti muncul Wizards, klik "Cancel" untuk membatalkannya.
2. Tambahkan dua buah control **Button**. Letakkan control tersebut pada posisi yang kira-kira cukup 'enak' untuk diklik kalau pemakai ingin menyimpan data.
3. Pasang gambar sebagai pengganti teks Caption yang biasa (lebih mudah dilihat oleh pemakai).

Gambar 6-26. Menambahkan tombol Simpan dan Undo

4. Tombol **Simpan** diberi nama **cmdSave**, dan tombol **Undo** diberi nama **cmdUndo**.

5. Jika dianggap perlu, kita bisa mengisi properti **ControlTip Text** agar ketika pemakai menunjuk tombol dengan mouse, muncul informasi fungsinya.

6.7 Menambahkan Event Procedure

Setelah selesai menambahkan control-control yang diperlukan pada form, sekarang kita akan menambahkan beberapa *event procedure* agar form bisa bereaksi terhadap aksi pemakai, misalnya ketika pemakai melakukan klik-ganda pada **Image** untuk memasang gambar, atau ketika pemakai memilih Jenis CD "Album" maka datasheet yang ditampilkan adalah dari frm_Konten_2, dan sebagainya.

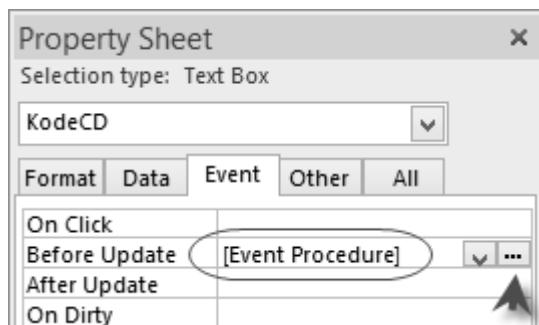
6.7.1 Menambahkan Function fRefreshList()

Hal pertama yang mudah bagi Anda adalah menambahkan function **fRefreshList()** pada control Combo Box, yaitu **Penerbit**, **FormatCD**, **Kategori**, dan **Jenis**. Caranya sama seperti yang Anda lakukan sebelumnya, yaitu mengopi function **fRefreshList()** dari salah satu module form yang mengandungnya, lalu mem-paste pada module **frm_Data**. Selanjutnya memasang **=fRefreshList()** pada event **On Enter** setiap Combo Box.

6.7.2 Menambahkan Event pada KodeCD

Hal kedua yang penting adalah melakukan validasi data yang dimasukkan pada field **KodeCD**. Walaupun field ini tidak kita jadikan Primary Key (karena tidak ingin memaksa pemakai memasukkan data sehingga bisa kosong), namun jika pemakai memasukkan Kode yang duplikat hendaknya diberi peringatan.

Sebenarnya validasi agar data yang dimasukkan unik bisa dilakukan pada tingkat table (pada Table Design), yaitu dengan mengatur properti **Indexed** = "Yes (no duplicates)". Namun jika ini dilakukan, peringatan baru muncul ketika record akan disimpan, dan peringatan tersebut berupa pesan error yang mungkin akan membuat pemakai bingung.

Untuk menambahkan event procedure yang dimaksud, klik field **KodeCD** lalu tampilkan **Property Sheet**. Pada event **Before Update**, klik tanda panah dan pilih "[Event Procedure]". Klik tombol *builder* ([...]) untuk menampilkan Visual Basic Editor.

Gambar 6-27. Memasukkan event procedure

Masukkan code sebagai berikut:

```
Private Sub KodeCD_BeforeUpdate(Cancel As Integer)
    If Not IsNull(Me.KodeCD) Then
        whr = "KodeCD='" & Me.KodeCD & "' AND CDID<>" &
Nz(Me.CDID, 0)
        If Not IsNull(DLookup("KodeCD", "tbl_CD", whr)) Then
            Alert "Kode CD '" & Me.KodeCD & _
                  "' sudah terpakai! Harap menggunakan Kode
yang lain."
```

```
            Cancel = True
        End If
    End If
End Sub
```

Secara sederhana code di atas bermaksud:

- Jika **KodeCD** kosong, maka tidak perlu melakukan apa-apa.
- Jika **KodeCD** berisi data, lakukan pengecekan apakah data tersebut ditemukan pada tbl_CD, yaitu pada record selain record sekarang. Dalam hal ini saya menggunakan *built-in function* DLookup().
- Jika data sudah ada, tampilkan pesan, lalu batalkan event Update.

Saya menggunakan event **Before Update** agar jika ternyata Kode CD sudah ada, update pada field bisa dibatalkan (Cancel = True) dan focus tidak pindah dari field **KodeCD**.

6.7.3 Membuat Procedure Alert()

Penampilan pesan pada procedure di atas menggunakan sebuah procedure buatan sendiri bernama **Alert()**.

Procedure **Alert()** disimpan pada sebuah module general yang saya beri nama **mdl_General**.

Buatlah sebuah module baru dengan klik "Module" pada ribbon tab **Create**. Simpan module dengan nama **mdl_General**.

Masukkan code sebagai berikut:

```
Sub Alert(pMsg)
    MsgBox pMsg & Space(5), vbCritical, vAppTitle
End Sub
```

Pada procedure di atas saya menggunakan sebuah *constant* bernama **vAppTitle.** Constant ini berisi nama atau judul aplikasi.

Untuk membuat constant ini, tekan **Ctrl+Home** pada module **mdl_General** untuk menuju ke baris paling atas. Di bawah pernyataan "Option Compare Database" masukkan deklarasi sebagai berikut:

```
Const vAppTitle = "CD Browser", vVersion = "1.0 (Agustus
2013)"
```

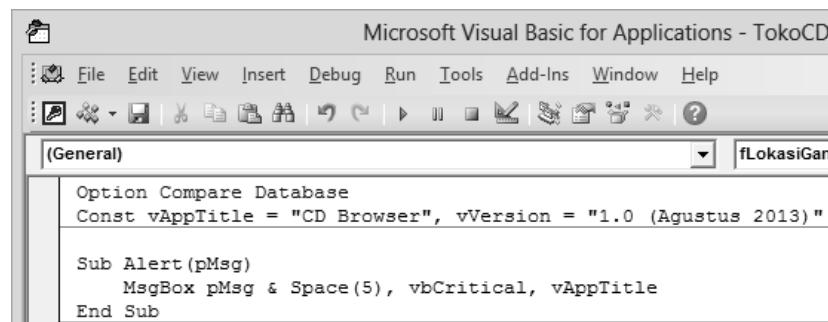
Pada kesempatan ini, kita sekaligus menambahkan constant **vVersion** yang berisi versi aplikasi. Kedua constant ini hampir selalu saya gunakan dalam program.

Microsoft Visual Basic for Applications - TokoCD

File Edit View Insert Debug Run Tools Add-Ins Window Help

(General) fLokasiGan

```
Option Compare Database
Const vAppTitle = "CD Browser", vVersion = "1.0 (Agustus 2013)"

Sub Alert(pMsg)
    MsgBox pMsg & Space(5), vbCritical, vAppTitle
End Sub
```

Gambar 6-28. Module mdl_General

6.7.4 Menambahkan Event pada Barcode

Kembali ke **frm_Data**, pada field **Barcode** kita perlu melakukan hal yang sama seperti pada field **KodeCD**, yaitu melakukan validasi data *barcode* yang dimasukkan pemakai untuk menghindari duplikasi. Jika data barcode ini duplikat (untuk CD yang berbeda) maka ketika Kasir melakukan *scanning* barcode, CD yang masuk ke dalam struk penjualan tidak pasti CD yang dimaksudkan pembeli!

Tambahkan code berikut ini pada event **Before Update** field **Barcode** (module form **frm_Data**):

```
Private Sub Barcode_BeforeUpdate(Cancel As Integer)
    If Not IsNull(Me.Barcode) Then
        whr = "Barcode='" & Me.Barcode & "' AND CDID<>" &
Nz(Me.CDID, 0)
        If Not IsNull(DLookup("KodeCD", "tbl_CD", whr)) Then
            Alert "Barcode '" & Me.Barcode & _
                  "' sudah terpakai! Harap memeriksa barcode."
            Cancel = True
        End If
    End If
End Sub
```

6.7.5 Menambahkan Event pada Kategori

Selanjutnya adalah menambahkan event pada combo box **Kategori**, selain event **On Enter** yang sudah dipasangi function **fRefreshList()**. Event yang akan kita tambahkan adalah event **After Update**, yaitu untuk menentukan klasifikasi/genre yang tepat sesuai kategori yang dipilih pemakai.

Perlu diperhatikan bahwa Combo Box yang akan dikenai oleh event ini adalah combo box **Klasifikasi** yang berada pada subform **Sub1**.

Tambahkan code berikut ini pada event **After Update** combo box **Kategori**:

```
Private Sub Kategori_AfterUpdate()
    tSql = "EXEC sp_Klasifikasi '" & Me.Kategori & "'"
    Me.Sub1!Klasifikasi.RowSource = tSql
End Sub
```

Pada procedure di atas dilakukan penggantian properti **Row Source** dari combo box **Klasifikasi** yang berada pada subform **Sub1**. Selain itu dilakukan pemanggilan procedure **Jenis_AfterUpdate** agar datasheet yang ditampilkan pada subform **Sub2** sesuai dengan kategori pilihan pemakai.

6.7.6 Menambahkan Event pada Jenis

Event selanjutnya yang perlu kita tambahkan adalah event **After Update** pada combo box **Jenis**. Event procedure ini akan mengganti form yang dijadikan **Source Object** pada subform **Sub2** sesuai "jenis" CD yang dipilih pemakai.

Tambahkan code berikut ini pada event **After Update** combo box **Jenis**:

```
Private Sub Jenis_AfterUpdate()
    If Nz(Me.Jenis, "Album") = "Kompilasi" Then
        Me.Sub2.SourceObject = "frm_Konten_1"
    Else
        Me.Sub2.SourceObject = "frm_Konten_2"
    End If
End Sub
```

6.7.7 Menambahkan Event pada Subform Sub1

Pada subform **Sub1** kita perlu menambahkan sebuah procedure pada event **On Enter**. Procedure ini dimaksudkan untuk mencegah pemakai 'masuk' ke

subform sebelum memasukkan data pada form utama. Perlu diketahui bahwa subform **Sub1** mengandalkan field **CDID** dari form utama untuk menyimpan data. Field **CDID** pada form utama akan terisi jika pemakai sudah memasukkan (dan menyimpan) data. Perhatikan properti **Link Master Fields** dan **Link Child Fields** pada subform **Sub1** yang mengaitkan form utama dan subform.

Tambahkanlah code berikut ini pada event **On Enter** subform **Sub1**:

```
Private Sub Sub1_Enter()
    If IsNull(Me.CDID) Then Me.txtDummy.SetFocus
End Sub
```

Pada procedure di atas, jika field **CDID** (pada form utama) kosong maka *focus* langsung dipindahkan ke control **txtxDummy**. Dengan demikian pemakai tidak bisa memasukkan/meng-update data pada subform.

6.7.8 Menambahkan Event pada Subform Sub2

Event yang perlu kita tambahkan pada subform **Sub2** sama dengan event pada subform **Sub1** di atas, yaitu event **On Enter**, dengan fungsi yang sama pula.

Tambahkanlah code berikut ini pada event **On Enter** subform **Sub2**:

```
Private Sub Sub2_Enter()
    If IsNull(Me.CDID) Then Me.txtDummy.SetFocus
End Sub
```

6.7.9 Menambahkan Event pada Image1

Selanjutnya adalah membuat sebuah trik untuk "memasang" gambar sampul (*cover*) CD pada control **Image1**. Ini memang sebuah trik karena di sini kita tidak mempunyai field untuk menyimpan data file gambarnya.

Catatan: Trik ini baru saya perkenalkan pada buku ini, sedangkan pada buku-buku lain saya menggunakan trik yang berbeda. Lihat, misalnya buku "Database Koperasi".

Ide atau skenarionya sebagai berikut.

1. Kita bisa menggunakan KodeCD sebagai nama file gambar, jadi tidak perlu menyimpan nama file gambar yang asli. Dengan cara ini, nama file gambar tidak akan duplikasi dan saling menimpa.

   Jika pemakai tidak memasukkan KodeCD, maka 'apa boleh buat' kita akan menggunakan data CDID yang pasti ada.

2. File gambar ditaruh pada sebuah lokasi di server yang bisa dijangkau oleh semua pemakai (*shared folder*). Dengan pengaturan pada Windows, kita bisa menyembunyikan folder penyimpanan gambar tersebut sehingga tidak terlihat oleh sembarang orang, dan mengatur *permission* agar hanya petugas yang berhak saja yang bisa *read-write,* sedangkan *everyone* hanya boleh *read* saja.

   Saya lebih menyarankan agar file gambar disimpan di luar database karena tidak membebani database serta mudah dalam pengelolaannya. Pengelolaan di sini antara lain:

   - Penyimpanan file gambar bisa dilakukan oleh admin secara langsung tanpa melalui program, karena hanya perlu menyimpan gambar di lokasi penyimpanan dengan nama sesuai Kode CD.
   - File gambar bisa ditampilkan dengan mudah menggunakan control Image, tidak perlu menangani gambar yang disimpan dalam database yang caranya lebih rumit.

3. File gambar bisa disimpan dalam empat format yang didukung, yaitu JPG atau JPEG, PNG, GIF, dan BMP. Program yang kita susun nanti akan mencari format JPG terlebih dahulu, lalu JPEG, PNG, GIF, baru BMP. Jika sudah menemukan sebuah format (berdasarkan ekstensi file) maka file yang ditemukan terlebih dahulu itulah yang digunakan.

4. Petugas bisa memilih file gambar dari lokasi mana saja (termasuk folder komputer lokal). Setelah file dipilih, program akan melakukan proses kopi file ke lokasi penyimpanan gambar. Jika file gambar sudah ada di lokasi penyimpanan (nama dan ekstensi sama), file gambar tersebut akan digantikan dengan file baru.

5. Informasi tentang **lokasi** file gambar disimpan dalam sebuah table. Table ini, seperti yang biasa saya lakukan, bernama **tbl_Config** (pembahasan tentang table ini akan dilakukan kemudian).

Ada tiga event pada control **Image1** yang akan kita buat, yaitu event **On Click**, event **On DblClick**, dan event **On Mouse Move**.

Event **On Click** digunakan oleh pemakai untuk menampilkan gambar asli, dalam hal ini kita akan menggunakan software **Image Viewer** default yang terinstal pada komputer.

Event **On DblClick** (*double-click*) digunakan untuk menampilkan kotak dialog **Open,** di mana petugas admin bisa memilih file gambar yang akan dipasang.

Event **On Mouse Move** digunakan untuk menampilkan ikon bergambar tangan jika mouse melewati control **Image1**.

Berikut ini event procedure **On Click** pada control **Image1**:

```
Private Sub Image1_Click()
    If Nz(Me.Image1.Picture, "(none)") <> "(none)" Then
        vHndl = fHandleFile(Me.Image1.Picture, WIN_MAX)
        If vHndl <> -1 Then
            Alert vHndl
        End If
    Else
        Image1_DblClick 0
    End If
End Sub
```

Pada procedure di atas, jika properti **Picture** pada **Image1** tidak sama dengan "(none)", yang berarti <u>ada</u> nama file gambar yang sedang digunakan, maka file gambar tersebut ditampilkan melalui function bernama **fHandleFile()**. Jika sama dengan "(none)", yang berarti <u>tidak ada</u> gambar yang digunakan, maka jalankan procedure **Image1_DblClick.** Parameter **0** hanya sekadar nilai saja (0 dan -1) karena dibutuhkan oleh procedure **Image1_DblClick.**

Catatan: function **fHandleFile()** akan dibahas kemudian.

Berikut ini event procedure **On DblClick** pada control **Image1**:

```
Private Sub Image1_DblClick(Cancel As Integer)
    If Not Me.AllowEdits Then
```

```
        Exit Sub
    End If
    '
    If IsNull(Me.CDID) Then
        Alert "Harap memasukkan/menyimpan data terlebih
dahulu!"
        Exit Sub
    End If
    '
    aFile = GetFileName(, , "Pilih File Gambar", , , gfnImage)
    If Nz(aFile, "") <> "" Then
        tFile = fLokasiGambar() & "\" & Nz(Me!KodeCD, Me!CDID)
& "." & FileExtension(aFile)
        FileCopy aFile, tFile
        '
        If IsFileExists(tFile) Then
            'hapus file sebelumnya, jika beda nama (daripada
nyampah)
            If tFile <> Me.Image1.Picture And
Me.Image1.Picture <> "(none)" Then
                DeleteFile Me.Image1.Picture
            End If
            'pasang picture
            Me.Image1.Picture = tFile
        Else
            'copy file tidak berhasil
            Me.Image1.Picture = ""
        End If
    End If
End Sub
```

Catatan: Pada procedure di atas digunakan beberapa function buatan sendiri, yaitu **GetFileName()**, **fLokasiGambar()**, **FileExtension()**, **IsFileExists()** dan **DeleteFile()**. Function-function tersebut akan kita bahas kemudian.

Untuk event **On Mouse Move** pada control **Image1** diisi dengan **=Tunjuk()**.

Function **Tunjuk()** digunakan untuk mengubah ikon mouse menjadi gambar tangan, yang mengindikasikan kepada pemakai bahwa control bisa diklik. Function ini akan dibahas tersendiri kemudian.

6.7.10 Menambahkan Event pada Tombol Save dan Undo

Event yang ditambahkan pada tombol **cmdSave** adalah event **On Click**, yaitu:

```
Private Sub cmdSave_Click()
    SaveRecord
End Sub
```

Event yang ditambahkan pada tombol **cmdUndo** adalah event **On Click**, yaitu:

```
Private Sub cmdUndo_Click()
    UndoRecord
End Sub
```

Adapun function **SaveRecord()** dan **UndoRecord()** adalah function buatan sendiri yang ditaruh pada **mdl_General**. Kedua function tersebut adalah sebagai berikut:

```
Public Function SaveRecord()
    On Error Resume Next
    DoCmd.RunCommand acCmdSaveRecord
End Function

Public Function UndoRecord()
    On Error Resume Next
    DoCmd.RunCommand acCmdUndo
End Function
```

6.7.11 Menambahkan Event pada Form

Selain event pada control, kita juga perlu memasang beberapa event procedure pada objek form. Untuk memasang event procedure berikut ini terlebih dahulu Anda harus memilih (*select*) objek **Form**.

Event procedure **On Current**:

```
Private Sub Form_Current()
    Me.txtDummy.SetFocus
    If Not IsNull(Me.CDID) Then 'hanya jika sdh ada record
        lok = fLokasiGambar()
        For N = 1 To 5
            fmt = Choose(N, "jpg", "jpeg", "png", "gif",
"bmp")
            fil = lok & "\" & Nz(Me!KodeCD, Me!CDID) & "." &
fmt
            If IsFileExists(fil) Then
                Me.Image1.Picture = fil
                Exit For
            Else
                Me.Image1.Picture = ""
            End If
        Next
    End If
    '
    Kategori_AfterUpdate
    Jenis_AfterUpdate
End Sub
```

Event **On Current** terjadi ketika form menampilkan sebuah record. Pada saat ini kita memasang *focus* ke **txtDummy** agar di saat pindah record, *focus* tidak tetap berada pada field sekarang dan langsung menyeleksi data pada field tersebut.

Selain itu, jika record sudah memiliki **CDID** (artinya record sudah ada, bukan record baru), program akan langsung mencari file gambar untuk record tersebut. Jika ditemukan, maka langsung dipasang. Jika tidak ditemukan, properti **Picture** perlu dikosongkan agar tidak tetap menampilkan gambar yang terpasang sebelumnya.

Event procedure **Form Resize**:

```
Private Sub Form_Resize()
    On Error Resume Next
    mrg = Me.Sub1.Left
    Me.Sub1.Height = Maxi(Me.InsideHeight - Me.Sub1.Top - mrg,
0)
    Me.Sub2.Height = Maxi(Me.InsideHeight - Me.Sub2.Top - mrg,
0)
    Me.Sub2.Width = Maxi(Me.InsideWidth - Me.Sub2.Left - mrg,
0)
    Me.lblDaftarIsi.Width = Me.Sub2.Width
End Sub
```

Event **Form Resize** terjadi ketika form diubah ukurannya. Event ini terjadi juga ketika form baru ditampilkan, walaupun tidak ada upaya *resize* oleh pemakai.

Apa yang dilakukan oleh procedure di atas adalah trik agar ukuran beberapa control disesuaikan dengan ukuran form. Dalam hal ini control yang kita ubah ukurannya (berdasarkan ukuran form) adalah subform **Sub1** dan subform **Sub2**. Subform **Sub1** diubah tingginya jika tinggi form berubah, sedangkan subform **Sub2** diubah tinggi dan lebarnya mengikuti perubahan tinggi dan lebar form.

Variable **mrg** digunakan untuk menyimpan ukuran yang dijadikan "margin", agar ruang kosong di sebelah kiri, kanan, dan bawah sama.

lblDaftarIsi adalah nama Label di atas subform **Sub1**. Kita perlu memberi nama sendiri pada Label tersebut agar lebih mudah dikenal. Pada procedure di atas, ukuran lebar Label diubah sesuai dengan ukuran lebar subform-nya.

Pada procedure di atas kita memasang perintah "On Error Resume Next" yang berfungsi: jika ada error maka jalankan perintah berikutnya, atau abaikan error. Perintah ini biasa saya gunakan pada procedure seperti ini untuk mengabaikan error yang mungkin muncul, karena tidak terlalu penting untuk dibuatkan *error handler*-nya. Error mungkin muncul, misalnya, jika pemakai secara ekstrem mengecilkan ukuran form sehingga nilai properti yang dipasang tidak bisa diterima (*not acceptable*).

Pada procedure di atas kita juga menggunaan function **Maxi()**. Function ini adalah fungsi buatan sendiri yang ditaruh pada **mdl_general**.

Tambahkanlah function **Maxi()** berikut ini pada **mdl_General**:

```
Function Maxi(n1, n2)
    If Nz(n1, 0) < Nz(n2, 0) Then
        Maxi = Nz(n2, 0)
    Else
        Maxi = Nz(n1, 0)
    End If
End Function
```

Function **Maxi()** digunakan untuk mencari nilai yang lebih tinggi dari dua nilai yang dikirimkan sebagai parameter.

6.8 Membuat Function fHandleFile()

Function **fHandleFile()** digunakan untuk membuka file eksternal (di luar database). Function ini dibuat oleh Dev Ashish, dan saya memperolehnya dari http://www.mvps.org/access/api/api0018.htm.

Anda tidak perlu mengetik sendiri function ini, tetapi langsung kopi saja dari file **mdl_fHandleFile.txt**, yang saya sediakan pada Bonus CD buku ini.

Cara melakukan kopi:

1. Buka file **mdl_fHandleFile.txt** dari Bonus CD.
2. **Select All** teks yang ada dalam file tersebut, lalu tekan **Ctrl+C**.
3. Pada Access Project, klik ribbon tab **Create**, klik "**Module**".
4. Di bawah baris pernyataan "Option Compare Database", tekan **Ctrl+V**.

5. Simpan module dengan nama, misalnya **mdl_fHandleFile**.

6.9 Membuat Function GetFileName()

Function **GetFileName()** digunakan untuk mencari/menelusuri letak file dengan kotak dialog dan mendapatkan nama filenya. Function ini dibuat oleh Graham R Seach (gseach@pacificdb.com.au).

Seperti pada function **fHandleFile()** di atas, Anda tidak perlu mengetik sendiri function **GetFileName()** ini. Saya sudah menyediakannya dalam Bonus CD dengan nama file **mdl_FileFolder.txt**. Kopilah dengan cara yang sama dan masukkan ke sebuah module, misalnya bernama **mdl_FileFolder**.

6.10 Membuat Function fLokasiGambar()

Function **fLokasiGambar()** adalah function kecil yang berfungsi mencari alamat lokasi penyimpanan gambar. Sebenarnya kita bisa saja menggunakan sebuah variable, namun dalam masa pengembangan aplikasi, pemakaian variable kadang-kadang mengharuskan kita menjalankan program dari awal lagi, misalnya ketika variable tersebut terhapus dari memori karena jalannya program dihentikan (**End**).

Tambahkanlah function **fLokasiGambar()** berikut ini ke dalam module **mdl_General**:

```
Function fLokasiGambar()
    tLoka = DLookup("LokasiGambar", "tbl_Config")
    If IsNull(tLoka) Then
        tLoka = CurrentProject.Path
    Else
        If Right(tLoka, 1) = "\" Then
            tLoka = Left(tLoka, Len(tLoka) - 1)
        End If
    End If
    fLokasiGambar = tLoka
End Function
```

Catatan: Pada function di atas, kita mengambil alamat lokasi gambar dari **tbl_Config**. Table ini akan kita buat kemudian.

6.11 Membuat Function FileExtension()

Function **FileExtension()** berfungsi untuk menentukan ekstensi dari sebuah nama file, misalnya untuk mengetahui apakah file tersebut berekstensi JPG, JPEG, dan sebagainya.

Tambahkanlah function **FileExtension()** sebagai berikut pada module **mdl_FileFolder**:

```
Function FileExtension(ByVal pFileName)
    If IsMissing(pFileName) Or IsNull(pFileName) Then Exit
Function
    p = InStrRev(pFileName, ".")
    FileExtension = Right(pFileName, Len(pFileName) - p)
End Function
```

6.12 Membuat Function IsFileExists()

Function **IsFileExists()** berfungsi untuk memeriksa keberadaan sebuah file berdasarkan *path* yang diberikan.

Tambahkanlah function **IsFileExists()** sebagai berikut pada module **mdl_FileFolder**:

```
Function IsFileExists(ByVal pFilePath) As Boolean
    Dim fso
    Set fso = CreateObject("Scripting.FileSystemObject")
    IsFileExists = fso.FileExists(pFilePath)
    Set fso = Nothing
End Function
```

6.13 Membuat Function DeleteFile()

Function **DeleteFile()** digunakan untuk menghapus sebuah file eksternal. Function ini akan mengembalikan nilai **True** jika penghapusan file berhasil dilakukan.

Tambahkanlah function **DeleteFile()** sebagai berikut pada module **mdl_FileFolder**:

```
Function DeleteFile(ByVal pFilePath) As Boolean
    If Not IsNull(pFilePath) Then
```

```
        If IsFileExists(pFilePath) Then
            On Error GoTo err_trap
            SetAttr pFilePath, vbNormal
            Kill pFilePath
            If Not IsFileExists(pFilePath) Then
                DeleteFile = True
            End If
        End If
    End If
err_trap:
End Function
```

6.14 Membuat Function Tunjuk()

Seperti telah dikemukakan di atas, function **Tunjuk()** digunakan untuk mengganti *mouse pointer* dengan gambar tangan (yang menunjuk).

Function ini sebenarnya hanya "shortcut" saja dari sebuah function lain yang bernama **MouseCursor()** yang dibuat oleh Terry Kreft.

Buatlah sebuah module baru dengan ribbon tab **Create**, "Module". Simpan module dengan nama, misalnya **mdl_Mouse**.

Buka file **mdl_Mouse.txt** yang saya sediakan pada Bonus CD, lalu kopi semua isinya ke **mdl_Mouse**.

Pada module **mdl_Mouse**, tambahkan function berikut ini:

```
Function Tunjuk()
    MouseCursor IDC_HAND
End Function
```

Simpan dan tutup module.

6.15 Membuat tbl_Config

tbl_Config adalah sebuah table yang digunakan untuk menyimpan nilai variable sehingga bisa digunakan setiap kali kita menjalankan program.

Pada saat ini, kita hanya perlu menyimpan tiga variable konfigurasi, yaitu Lokasi file gambar, Nama Toko, dan Alamat Toko.

Buatlah sebuah table dengan Table Design, dengan struktur sebagai berikut.

Gambar 6-29. Desain tbl_Config

Simpan table dengan nama **tbl_Config**.

Bukalah **tbl_Config** dalam **Datasheet View** agar Anda bisa mengisi data konfigurasi yang kita perlukan.

Pada contoh saya:

NamaToko: Toko CD "DISK RARA"

AlamatToko: Mall Ambasador Lt. 2 Blok B No. 5, Jakarta

LokasiGambar: D:\Programs\TokoCD\Images

Jumlah record pada **tbl_Config** cukup satu saja, yaitu yang kita tambahkan sekarang. Adapun nilai pada field **RecID** biarkan terisi otomatis dan tidak masalah nilainya berapa.

Untuk field **LokasiGambar**, Anda bisa mengisinya dengan format UNC (Universal Naming Convention) yang sering dipakai pada Windows, misalnya:

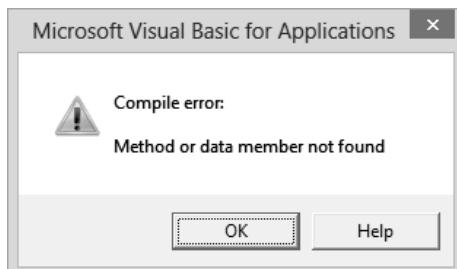
```
\\Netfile\Database\TokoCD$
```

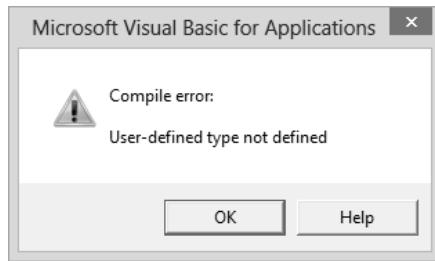
6.16 Melakukan Kompilasi

Hingga tahap ini, pembuatan form data yang Anda lakukan hampir selesai. Hal berikutnya yang perlu dilakukan adalah memastikan semuanya sudah benar. Cara pertama yang mudah adalah melakukan **Compile**.

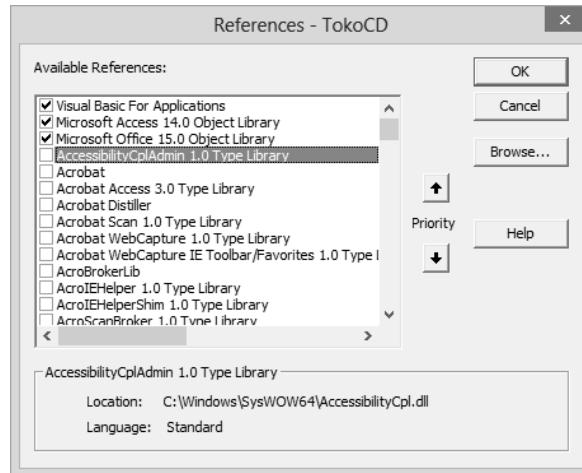
1. Tampilkan **Visual Basic Editor**. Shortcut key-nya: **Alt+F11**.
2. Pada jendela VBA, klik menu **Debug > Compile**.

Microsoft Access akan melakukan proses kompilasi, dan jika ada error akan langsung berhenti. Anda harus melakukan koreksi sehingga error tersebut hilang. Lakukan **Compile** kembali hingga tidak ada lagi error yang muncul.

Error yang dideteksi oleh **Compile** disebut *compilation error*. Penyebabnya bisa karena "Syntax Error" (pengetikan perintah yang salah), "Variable Not Found" (jika menggunakan Option Explicit), atau "Method or Data Member Not Found" (salah penulisan nama field atau nama properti).

Gambar 6-30. Error Method or data member not found

Jika Anda mengopi code dari Internet atau program lain, kemungkinan juga muncul error sebagai berikut.

Gambar 6-31. Error User-defined type not defined

Error ini mungkin disebabkan oleh **References** yang digunakan. Untuk memeriksa References, klik menu **Tools < References**.

Pada komputer saya, References yang digunakan sebagai berikut. Jika salah satu Reference tersebut saya *uncheck*, akan muncul error di atas.

Gambar 6-32. References

Catatan: Reference yang tersedia tergantung pada library (umumnya berupa file .dll) dari berbagai software yang terinstal pada komputer. References yang tersedia pada komputer saya mungkin berbeda dengan yang ada pada komputer Anda. Jadi, jika Anda mengopi code dari sumber lain dan muncul error "type not defined" atau yang semacam itu, cari tahulah References yang dibutuhkan.

Sampai pada tahap ini, form data yang Anda buat sudah selesai. Cobalah menjalankannya dan memasukkan data CD koleksi Anda.

Kopi file **TokoCD-frm_Data.adp** dari Bonus CD buku ini ke folder latihan Anda. Buka file tersebut (menggunakan *instance* Microsoft Access yang lain. Anda bisa menjalankan beberapa *instance* Microsoft Access sekaligus) dan lakukan koneksi ke database TOKOCD pada SQL Server Anda. Form **frm_Data** yang tersedia pada file **TokoCD-frm_Data.adp** seharusnya sama dengan yang Anda buat hingga tahap ini.

Gambar 6-33. Tampilan frm_Data